2 Getting Started with Python

Learning Outcome

<u>Coding Standards</u>
- Use indentation and white space
- Use naming conventions (e.g. meaningful identifier names)
- Write comments (name of programmer, date written, program description and version book-keeping/control)

<u>Programming Elements and Constructs</u>
- Understand the different data types: integer, real, char, string and Boolean
- Use common library functions for input/output, strings and mathematical operations

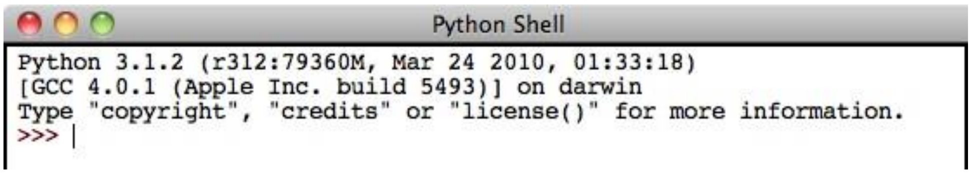
<u>Character Encoding</u>
- Give examples of where and how Unicode is used
- Use ASCII code in programs

Python is a high-level, general-purpose programming language for solving problems on modern computer systems. The official online resources can be found at www.python.org, and its popularity has generated many websites and YouTube channels for self-learning.

Three tools that we can use to write a program and test it.

Python Shell

After downloading Python, launch the IDLE to open a Python shell. Shell is useful for experimenting with short expressions or statements to learn new features of the language.

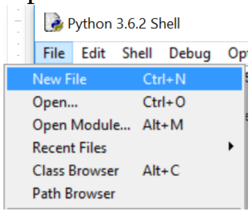


```
Python 3.1.2 (r312:79360M, Mar 24 2010, 01:33:18)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> |
```

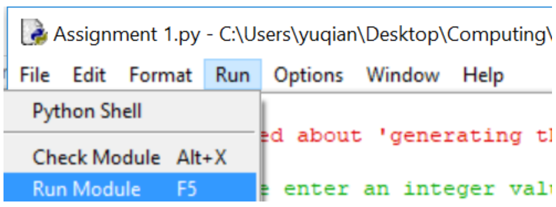
Python File

When we need to construct complex programs and test them, we can create and save files in program libraries to reuse or share with others.

Open a New File in Shell and save the program files use .py extention.

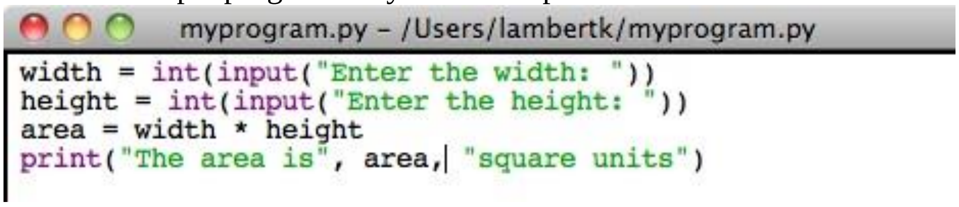


We can then run Python program files within IDLE using menu option F5 for Windows.


Assignment 1.py - C:\Users\yuqian\Desktop\Computing\
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5

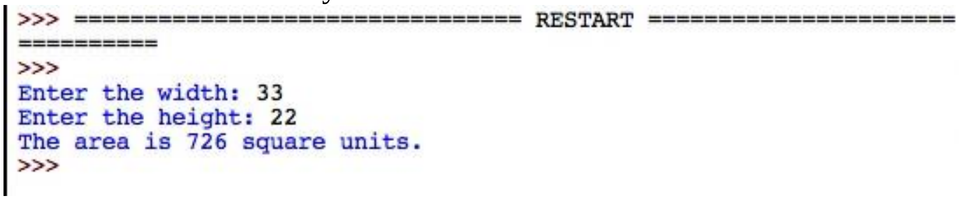
```
ed about 'generating tl
e enter an integer valu
```

Here is a sample program for your first experiment.



```
width = int(input("Enter the width: "))
height = int(input("Enter the height: "))
area = width * height
print("The area is", area, "square units")
```

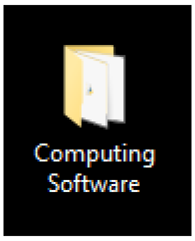
Then we run the file in Python Shell.



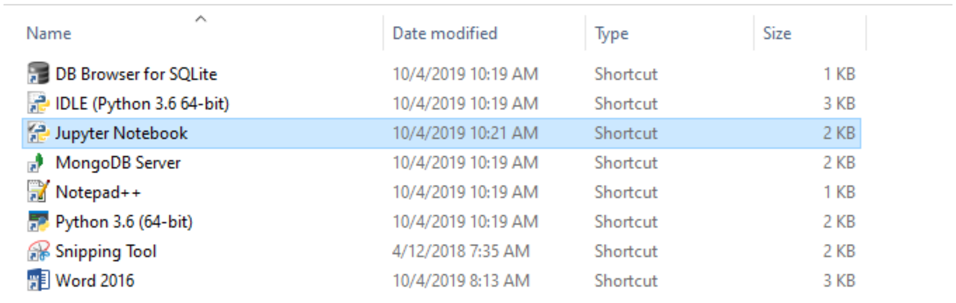
```
>>> ================================ RESTART ======================
==========
>>>
Enter the width: 33
Enter the height: 22
The area is 726 square units.
>>>
```

Running Jupyter Notebook on the school laptop

Look for Computing Software shortcut (shown below) on the desktop. Double click on the shortcut to open Windows Explorer.



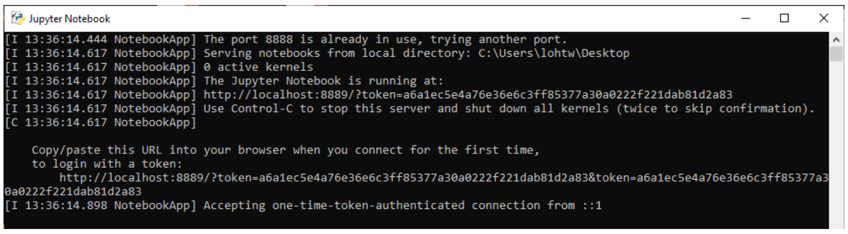
In Windows Explorer, a list of shortcuts is listed. Double click on the Jupyter Notebook shortcut to launch the application.



Name	Date modified	Type	Size
DB Browser for SQLite	10/4/2019 10:19 AM	Shortcut	1 KB
IDLE (Python 3.6 64-bit)	10/4/2019 10:19 AM	Shortcut	3 KB
Jupyter Notebook	10/4/2019 10:21 AM	Shortcut	2 KB
MongoDB Server	10/4/2019 10:19 AM	Shortcut	2 KB
Notepad++	10/4/2019 10:19 AM	Shortcut	1 KB
Python 3.6 (64-bit)	10/4/2019 10:19 AM	Shortcut	2 KB
Snipping Tool	4/12/2018 7:35 AM	Shortcut	2 KB
Word 2016	10/4/2019 8:13 AM	Shortcut	3 KB

A console windows will be launch prior to the browser. Do not close the console windows.

Take note that the console windows should remain open for Jupyter Notebook to work.



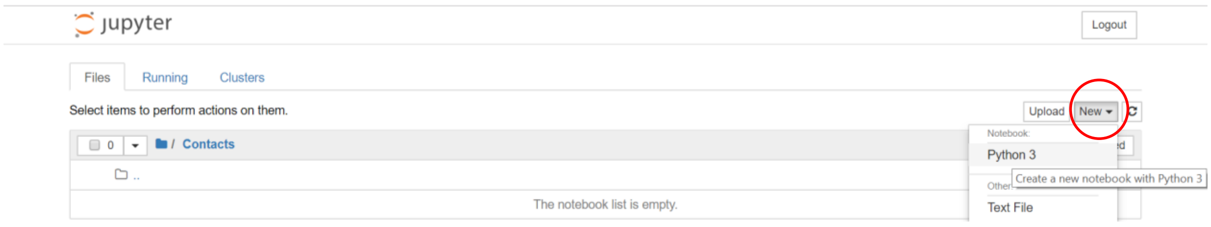
```
[I 13:36:14.444 NotebookApp] The port 8888 is already in use, trying another port.
[I 13:36:14.617 NotebookApp] Serving notebooks from local directory: C:\Users\lohtw\Desktop
[I 13:36:14.617 NotebookApp] 0 active kernels
[I 13:36:14.617 NotebookApp] The Jupyter Notebook is running at:
[I 13:36:14.617 NotebookApp] http://localhost:8889/?token=a6a1ec5e4a76e36e6c3ff85377a30a0222f221dab81d2a83
[I 13:36:14.617 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:36:14.617 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8889/?token=a6a1ec5e4a76e36e6c3ff85377a30a0222f221dab81d2a83&token=a6a1ec5e4a76e36e6c3ff85377a3
0a0222f221dab81d2a83
[I 13:36:14.898 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

Jupyter notebook home page will be shown in the browser. By default, the home page will display the files on the desktop.

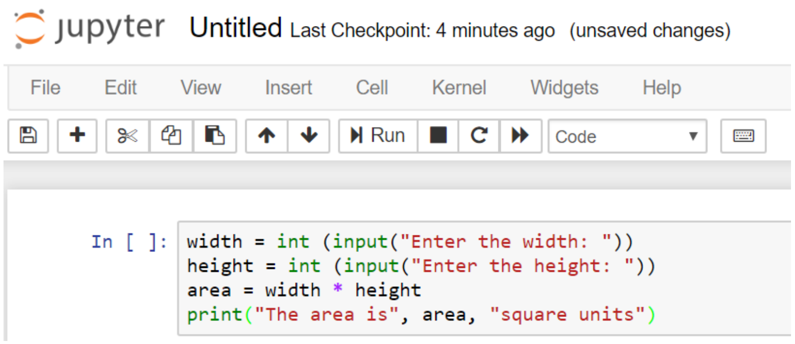
Jupyter Notebook File

These files are saved in .ipynb extension.

Click 'New' to create a new notebook in Python 3.

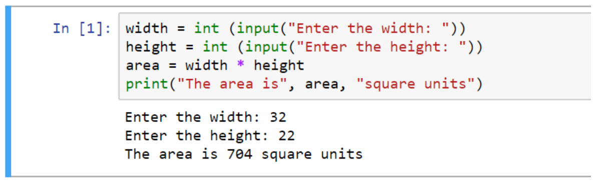

jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/ Contacts
..
The notebook list is empty.
Notebook:
Python 3
Create a new notebook with Python 3
Other:
Text File

Type the codes and run it by clicking the 'Run' button, or pressing 'Ctrl' + 'Enter'.


jupyter Untitled Last Checkpoint: 4 minutes ago (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
In [ ]: width = int (input("Enter the width: "))
        height = int (input("Enter the height: "))
        area = width * height
        print("The area is", area, "square units")
```

Here is the output.



```
In [1]: width = int (input("Enter the width: "))
        height = int (input("Enter the height: "))
        area = width * height
        print("The area is", area, "square units")

        Enter the width: 32
        Enter the height: 22
        The area is 704 square units
```

The convenience of using Jupyter Notebook is that we can write multiple programs in one file, by clicking the '+' button.

2.1 Comments and Docstrings

Programs should be easily understood and modifiable by any users, so we need to insert comments to explain how we solve the problem and sometimes the meaning of the variable names we defined.

Docstring for a paragraph of comments:

```
"""
Program: circle.py
Author: Ken Lambert
Last date modified: 7/10/08

The purpose of this program is to compute the area of a circle.
The input is an integer or floating-point number representing the
radius of the circle. The output is a floating-point number
labeled the area of the circle.
"""
```

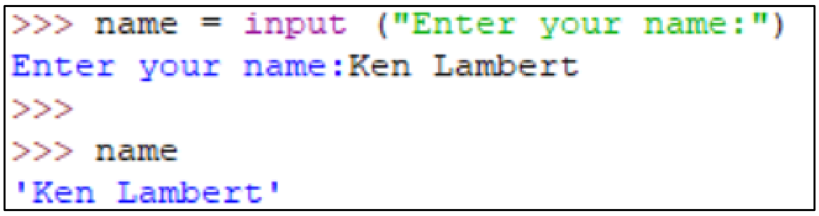
For short explanation, we use end-of-line comment with #:

```
>>> RATE = 0.85   # Conversion rate for Canadian to US dollars
```

2.2 Input and Output

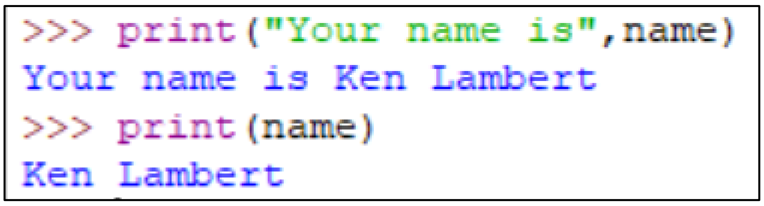
Programs usually accept inputs from a source, process them, and output results to a destination. In terminal-based interactive programs like Python, these are the keyboard and terminal display, respectively.

To get **input** from keyboard, we use: <variable identifier> = input (<a string prompt>)



```
>>> name = input ("Enter your name:")
Enter your name:Ken Lambert
>>>
>>> name
'Ken Lambert'
```

To generate the **output**, we use: print(<expression>)



```
>>> print("Your name is",name)
Your name is Ken Lambert
>>> print(name)
Ken Lambert
```

Question: why is it different for *name* and *print(name)*?

2.3 Data Types

In real life, we take many things as data, for example, integers, decimal numbers, characters, words, paragraphs. However, in programming, we need to define and categorize them properly.

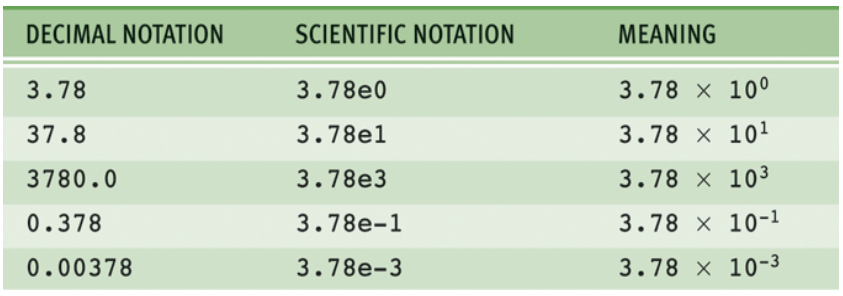
A data type consists of a set of values, a set of operations that can be performed on the values and the way values can be stored on computers. Here we learn the Python way of defining data types though other programming languages may have different names and categories.

2.3.1 Integers

- In real life, the range of integers is infinite.
- Computer's memory places a limit on largest magnitude of integers
- Python's **int** typical range is -2^{31} to $2^{31} - 1$

2.3.2 Floating-Point Numbers

- Python uses floating-point numbers to represent real numbers
- Python's **float** typical range: -10^{308} to 10^{308}
- Typical precision: 16 digits



DECIMAL NOTATION	SCIENTIFIC NOTATION	MEANING
3.78	3.78e0	3.78×10^{0}
37.8	3.78e1	3.78×10^{1}
3780.0	3.78e3	3.78×10^{3}
0.378	3.78e-1	3.78×10^{-1}
0.00378	3.78e-3	3.78×10^{-3}

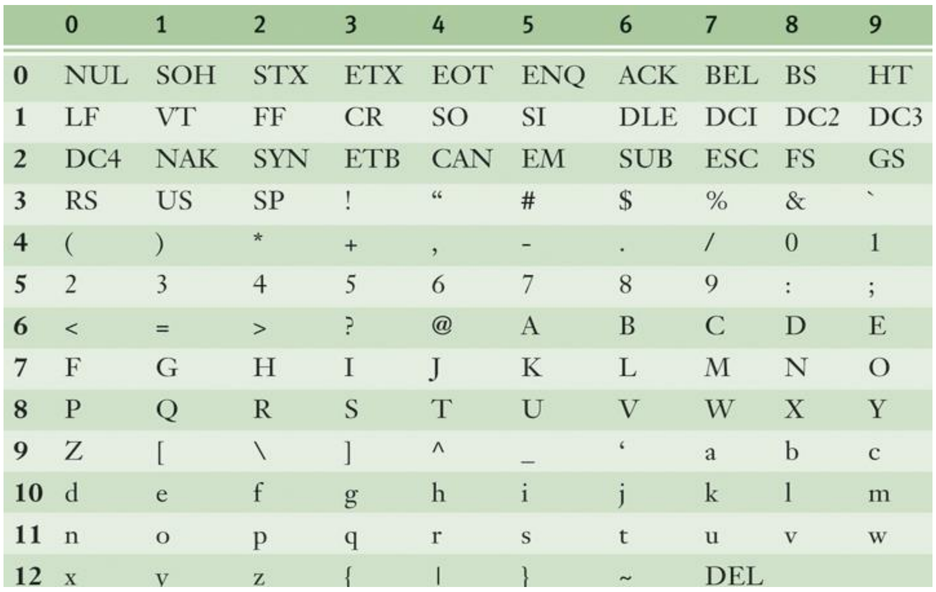
2.3.3 Characters and Strings

Characters in programming is not limited to the 26 English characters. It can also be space character, question mark, Chinese character, etc.

A **string** is a sequence of characters. It can be a word, a sentence or a paragraph.

Text processing is by far the most common application of computing. E-mail, text messaging, Web pages, and word processing all rely on and manipulate data consisting of strings of characters. But computer only reads zeros and ones, so character sets are invented to use numbers to represent English characters and notations we use. Unicode set and ASCII set are common character sets in use. A **bit** is a **bi**nary digi**t**, it can be a **1** or **0**. So if we use seven bits, we may have $2^7 = 128$ possible binary representation of numbers 0 - 127.

The original **ASCII** used seven bits to encode 128 different characters, as shown below.



	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DCI	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	SP	!	“	#	$	%	&	`
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	‘	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

In Python, we use **ord** and **chr** to convert characters to and from ASCII.

```
>>> ord('a')
97
>>> ord('A')
65
>>> chr(65)
'A'
>>> chr(66)
'B'
>>>
```

There is surely no need to memorize the table but we might use it to traverse within all the alphabets. For example, `chr(ord('A')+5)` gives ‘F’.

ASCII was further developed to eight bits to accommodate more characters. However, using more digits takes more memory space as well.

Unicode was developed to use a variable bit encoding program and Unicode standard defines UTF-8, UTF-16 and UTF-32. The large capacity of Unicode makes many non-English languages available, e.g. Chinese and Japanese. Furthermore, ASCII is compatible with Unicode.

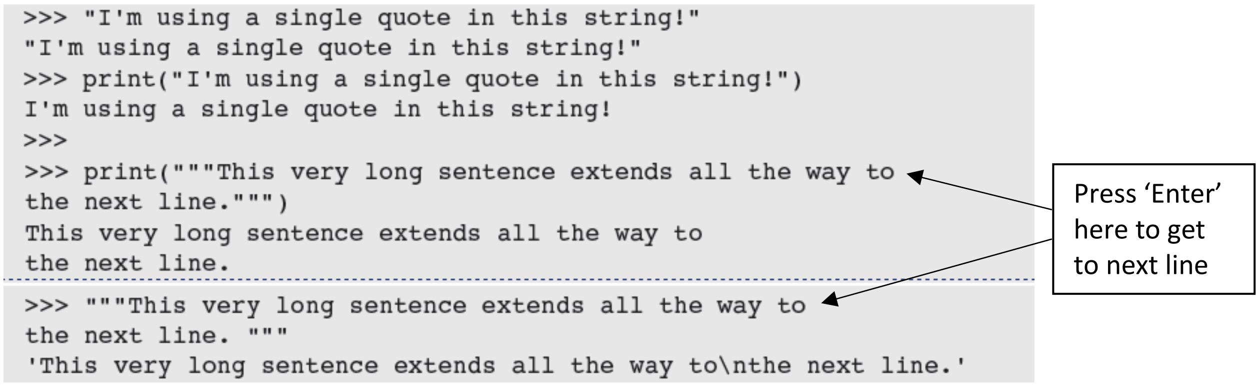
2.3.4 Boolean

Boolean data type consists of two values: **True** and **False**

2.4 Literal

TYPE OF DATA	PYTHON TYPE NAME	EXAMPLE LITERALS
Integers	int	-1, 0, 1, 2, 3420000556008
Real numbers	float	-0.55, .333, 3.14, 6.0
Character strings	str	“Hi”, “ “, ‘A’, ’66’

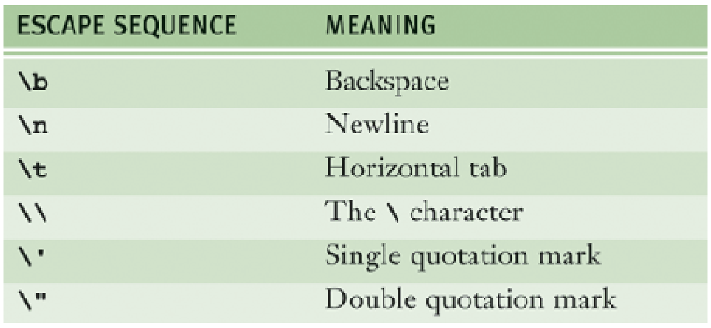
- A literal is the way a value of a data type looks to a programmer.
- In Python, a string literal is a sequence of characters enclosed in single or double quotation marks
- '' and "" represent the **empty string**
- Use ''' and """ for multi-line paragraphs



```
>>> "I'm using a single quote in this string!"
"I'm using a single quote in this string!"
>>> print("I'm using a single quote in this string!")
I'm using a single quote in this string!
>>>
>>> print("""This very long sentence extends all the way to
the next line.""")
This very long sentence extends all the way to
the next line.
```

```
>>> """This very long sentence extends all the way to
the next line. """
'This very long sentence extends all the way to\nthe next line.'
```

2.5 Escape Sequences



ESCAPE SEQUENCE	MEANING
`\b`	Backspace
`\n`	Newline
`\t`	Horizontal tab
`\\`	The \ character
`\'`	Single quotation mark
`\"`	Double quotation mark

2.6 Variables and the Assignment Statement

A variable associates a name with a value to make it easy to remember and use later. Variable naming rules:

– **Reserved words** cannot be used as variable names. Examples: if, def, import
– Name must **begin with** a letter or _
– Name can contain any number of **letters, digits, or _**
– Names are **case sensitive**. Example: WEIGHT is different from weight
– All uppercase letters for symbolic constants. Examples: TAX_RATE
– "camel casing". Example: InterestRate
– **Short names** are preferred. If necessary, we can write comments to explain short forms.

Variables receive initial values, and can be reset to new values with an **assignment statement.**

```
<variable name> = <expression>
```

Subsequent uses of the variable name in expressions are known as **variable references.**

```
>>> firstName = "Ken"
>>> secondName = "Lambert"
>>> fullName = firstName + " " + secondName
>>> fullName
'Ken Lambert'
>>>
```

2.7 Expressions

- A literal evaluates to itself
- A variable reference evaluates to the variable’s current value
- **Expressions** provide easy way to perform operations on data values to produce other values
- When entered at Python shell prompt, expression’s operands are evaluated and its operator is then applied to these values to compute the value of the expression

2.7.1 Arithmetic Expressions

An arithmetic expression consists of operands and operators combined in a manner that is already familiar to you in mathematics.

OPERATOR	MEANING	SYNTAX
**	Exponentiation	a ** b
−	Negation	− a
*	Multiplication	a * b
/	Division	a / b
//	Quotient	a // b
%	Remainder or modulus	a % b
+	Addition	a + b
−	Subtraction	a − b

Precedence rules of operators:

- ** has the highest precedence and is evaluated first
- Unary negation is evaluated next
- *, /, //, and % are evaluated before + and −
- + and − are evaluated before =
- With two exceptions, operations of equal precedence are left associative, so they are evaluated from left to right (** and = are right associative)
- You can use () to change the order of evaluation

EXPRESSION	EVALUATION	VALUE
5 + 3 * 2	5 + 6	11
(5 + 3) * 2	8 * 2	16
6 % 2	3	0
2 * 3 ** 2	2 * 9	18
− 3 ** 2	− (3 ** 2)	−9

2**3 **2	2 ** 9	512
(2** 3) ** 2	8 ** 2	64
45 / 0	Error: cannot divide by 0	
45 % 0	Error: cannot divide by 0	
3.14*3**2	3.14*(3**2)	28.26

Quotient VS Division:

```
3 // 2 * 5.0 yields 1 * 5.0, which yields 5.0
```

```
3 / 2 * 5 yields 1.5 * 5, which yields 7.5
```

2.7.2 Augmented Assignment

Standard formatting

```
<variable> = <variable> <operator> <expression>
```

can be shortened to

```
<variable> <operator>= <expression>
```

Augmented assignment operations:

```
a = 17
s = "hi"

a += 3          # Equivalent to a = a + 3
a -= 3          # Equivalent to a = a - 3
a *= 3          # Equivalent to a = a * 3
a /= 3          # Equivalent to a = a / 3
a %= 3          # Equivalent to a = a % 3
s += " there"   # Equivalent to s = s + " there"
```

2.7.3 String Concatenation

You can join two or more strings to form a new string using the concatenation operator +. The * operator allows you to build a string by repeating another string a given number of times

```
>>> ' ' * 10 + "Python"
'          Python'
>>>
```

2.7.4 Type Conversions

Sometimes, we need to convert between data types for calculation or formatting.

CONVERSION FUNCTION	EXAMPLE USE	VALUE RETURNED

int(<a number or a string>)	`int(3.77)`
	`int("33")`
		3
		33

float(<a number or a string>)	`float(22)`	22.0
str(<any value>)	`str(99)`	'99'

Note that the **int** function converts a **float** to an **int** by truncation, not by rounding.

```
>>> int(6.75)
6
>>> round(6.75)
7.0
>>> int(round(6.75))
7
```

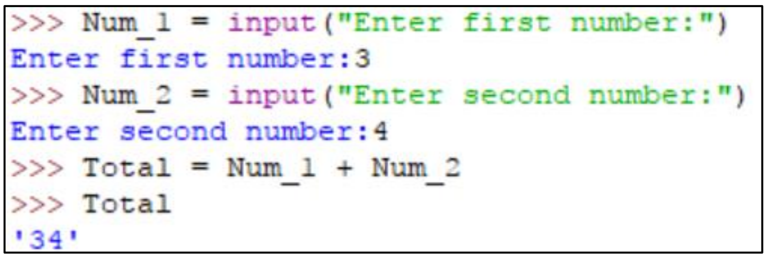
- Construction of strings from **numbers** and other **strings**.

```
>>> profit = 1000.55
>>> print('$' + profit)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'float' objects
```

Solution: use str function

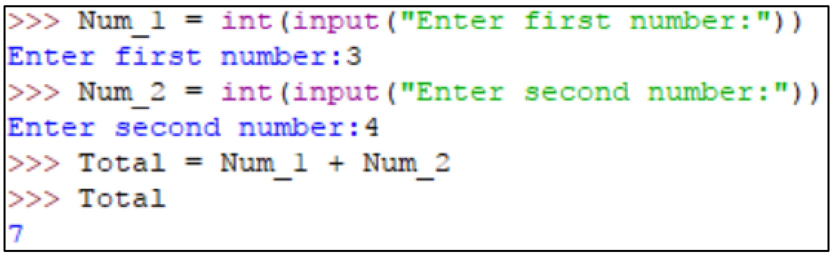
```
>>> print('$' + str(profit))
$1000.55
```

- Input data type is string by default.



```
>>> Num_1 = input("Enter first number:")
Enter first number:3
>>> Num_2 = input("Enter second number:")
Enter second number:4
>>> Total = Num_1 + Num_2
>>> Total
'34'
```

Solution: use **int** or **float** function



```
>>> Num_1 = int(input("Enter first number:"))
Enter first number:3
>>> Num_2 = int(input("Enter second number:"))
Enter second number:4
>>> Total = Num_1 + Num_2
>>> Total
7
```

2.8 Formatting Text for Output

- Many data-processing applications require output that has **tabular format**
- **Field width**: Total number of data characters and additional spaces for a datum in a formatted string

Using f-string formatting

- Placing between the quotation marks after the '**f**' the text that you want displayed
- Enclosing the variables to be displayed within the text in curly braces
- Within those curly braces, placing a colon (**:**) after the variable
- Formatting the variable using a format specification (width, alignment, data type) after the colon

```python
# Python f-string formatting

qty = 6             # integer
item = 'bananas'    # string
price = 1.75        # float

print (f'{qty} {item} cost ${price}')  #  6 bananas cost $1.75
```

```
6 bananas cost $1.75
```

```python
# Python f-string format width & alignment

i = 12345
s = 'abcde'

print (f"{i:<8}|{s:<8}")   # left aligned in a field width of 8
print (f"{s:>8}|{i:>8}")   # right aigned in a field width of 8

```

```
12345   |abcde
   abcde|   12345
```

```python
# Python f-string format floats

# {:<field width>.<precision>f}

x = 78.2348

print (f"{x:.3f}")     # 78.235

print (f"{x:<8.2f}")  # 78.23
print (f"{x:>8.2f}")  #    78.23

# No precision specified,python auto-fill up to 6 d.p.
print (f"{x:<12f}")   # 78.234800
print (f"{x:>12f}")   #    78.234800

```

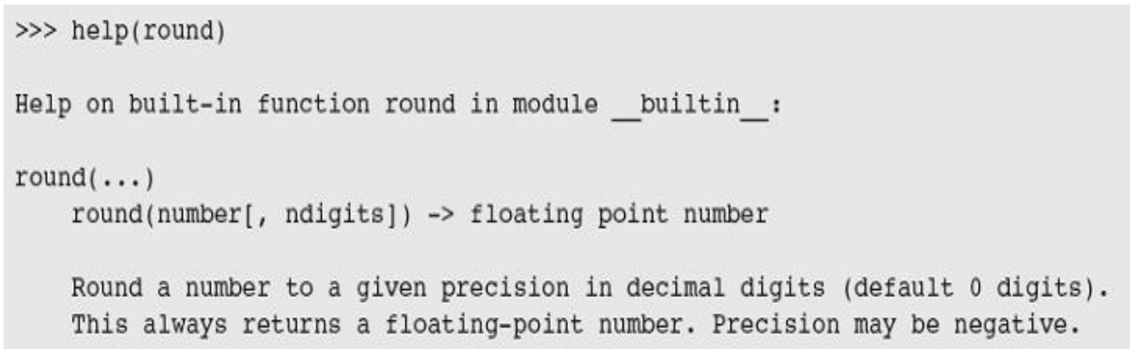
```
78.235
78.23
   78.23
78.234800
   78.234800
```

2.9 Using Functions and Modules

Python includes many useful functions, which are organized in libraries of code called **modules**.

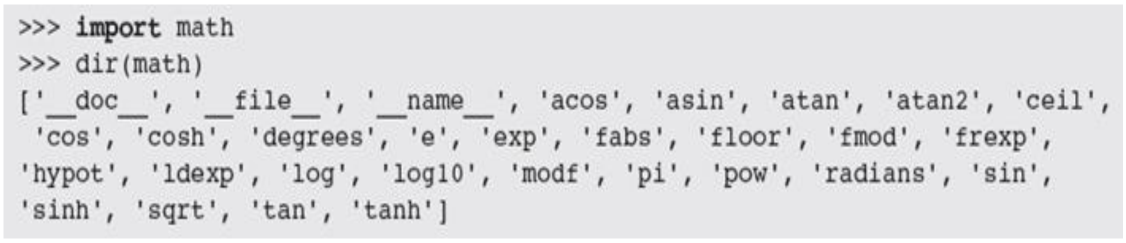
2.9.1 Calling Functions: Arguments and Return Values

- A function is chunk of code that can be called by name to perform a task
- Functions often require arguments or parameters
- When function completes its task, it may return a value back to the part of the program that called it



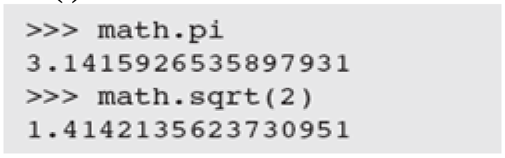
```
>>> help(round)

Help on built-in function round in module __builtin__:

round(...)
    round(number[, ndigits]) -> floating point number

    Round a number to a given precision in decimal digits (default 0 digits).
    This always returns a floating-point number. Precision may be negative.
```

2.9.2 The math Module



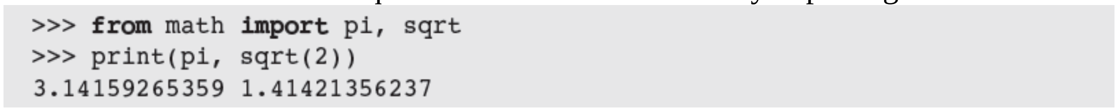
```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__name__', 'acos', 'asin', 'atan', 'atan2', 'ceil',
 'cos', 'cosh', 'degrees', 'e', 'exp', 'fabs', 'floor', 'fmod', 'frexp',
'hypot', 'ldexp', 'log', 'log10', 'modf', 'pi', 'pow', 'radians', 'sin',
'sinh', 'sqrt', 'tan', 'tanh']
```

To use a resource from a module, you write the name of a module as a qualifier, followed by a dot (.) and the name of the resource.



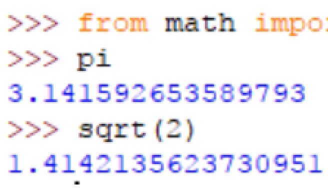
```
>>> math.pi
3.141592653589793l
>>> math.sqrt(2)
1.4142135623730951
```

You can avoid the use of the qualifier with each reference by importing the individual resources



```
>>> from math import pi, sqrt
>>> print(pi, sqrt(2))
3.14159265359 1.41421356237
```

You may import all of a module's resources to use without the qualifier.

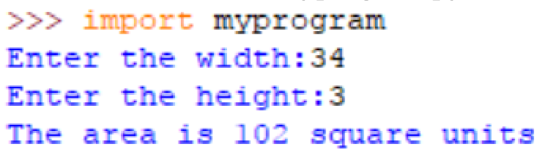


```
>>> from math import *
>>> pi
3.141592653589793
>>> sqrt(2)
1.4142135623730951
```

2.9.3 The Main Module

Like any module, the main module can be imported.

We save the first file myprogram.py in the libraries of Python, and it can be imported directly.



```
>>> import myprogram
Enter the width:34
Enter the height:3
The area is 102 square units
```

2.9.4 Program Format and Structure

- Start with comment in the form of a docstring, including author's name, purpose of program, and other relevant information
- Then, include statements that:
  - Import any modules needed by program
  - Initialize important variables, suitably commented
  - Prompt the user for input data and save the input data in variables
  - Process the inputs to produce the results
  - Display the results

Tutorial 2

1. Let the variable **x** be "dog" and the variable **y** be "cat". Write the values returned by the following operations:

Operations	Values Returned
x + **y**	
"the" + **x** + "chase the" + **y**	
x*4	
print(**x** + **y**)	
print(**x**,**y**)	

2. Test the output below for **round** and **int**.

Code	Output
round(10.6)	
int(10.6)	
round(10.666,2)	

3. Assume that the variable **amount** refers to **24.325**. Write the output of the following statements:

(a) `print (f"Your salary is ${amount:.2f}")`

(b) `print (f"The area is {amount:.1f}")`

(c) `print (f"{amount:7f}")`

(d) `print (f"BMI btw {18.5:f} and {amount:2.3f} is ideal")`

(e) `print (f"Overweight  ={amount:>7.2f} - {29.9:.2f}")`

4. Write a code segment that displays the values of the integers **x**, **y**, and **z** on a single line, such that each value is right-justified in 6 columns.

5. The **math** module includes a **pow** function that raises a number to a given power. The first argument is the number and the second argument is the exponent. Write a code segment that imports this function and calls it to print the value 8^2.

Assignment 2

1. Write a program that accepts the user’s name (as text) and age (as an integer) as input. The program should output a sentence containing the user’s name and current age, and age in 10 years’ time.

2. Write a program that takes the radius of a sphere (a floating-point number) as input and output the sphere’s diameter, circumference, surface area, and volume.

3. An employee’s total weekly pay equals the hourly wage multiplied by the total number of regular hours plus overtime pay. Overtime pay equals the overtime hours multiplied by 1.5 times the hourly wage. Write a program that takes as inputs the hourly wage, total regular hours, and total overtime hours and display an employee’s total weekly pay (round to the nearest dollar).

4. A bakery sells loaves of bread for $3.50 each. Day old bread is discounted by 60 percent. Write a program that begins by reading the number of loaves of day old bread being purchased from the user. Then your program should display the regular price for the bread, the discount because it is a day old, and the total price.

   All of the values should be displayed using two decimal places, and the decimal points in all of the numbers should be aligned when reasonable values are entered by the user.